Description

Method for decoding a data sequence encoded with the aid of a binary convolutional code

The present invention relates to a method for decoding a data sequence which consists of K information bits and has been encoded with the aid of a binary convolutional code, using a MaxLogMAP algorithm.

Voice channels and data channels of a radio communications system, which is designed to operate to the GSM/EDGE mobile radio communications standard for example, use binary convolutional codes for data coding and data decoding. A preferred algorithm for what is known as "soft input/soft output decoding" is the known "symbol-by-symbol log-likelihood maximum a posteriori probability" algorithm (LogMAP algorithm), which is generally implemented with the aid of a maximum approximation (MaxLogMAP algorithm).

The basic MAP algorithm is described, for example, in the publication "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", L.R. Bahl et al., IEEE Transactions on Information Theory, pp. 284-287, March 1974. The MaxLogMAP algorithm can be found in the publication "Iterative Decoding of Binary Block and Convolutional Codes", J. Hagenauer et al., IEEE Transactions on Information Theory, vol. 42, no. 2, pp. 429-445, March 1996.

A window MaxLogMAP algorithm implemented with the aid of what is known as a "sliding window" (decoding window) is described in the publication "An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes", A. J. Viterbi, IEEE Journal on Selected Areas in Communications, vol. 16, no.2, pp. 260-264, Feb. 1998.

The basis for decoding a data sequence encoded with the aid of a binary convolutional code is the binary trellis diagram. One segment

of the trellis diagram belonging to one information bit of the data sequence detects all possible combinations of m (convolutional code memory length) preceded information bits as 2^m initial states. Furthermore, all ensuing "conversions" (codings) of the information bit are detected as $2^{(m+1)}$ state transitions, and resulting 2^m target states are detected as initial states for the next consecutive information bit. Here an information bit sequence corresponds to a specific path within the trellis diagram, a sequence of the most probable information bits in the trellis diagram being established with the aid of the MaxLogMAP algorithm.

When the MaxLogMAP algorithm is implemented, a distinction essentially has to be made between a systolic implementation and a processor-oriented implementation.

With the systolic implementation the aim is to achieve as high a degree of parallelism as possible of decoding steps throughout the trellis diagram. This implementation is used with extremely high data throughput requirements of up to 50 Gbit/s.

Processor-oriented implementation is suitable for moderate data throughput requirements of a few Mbit/s using low-cost hardware.

For both implementations it is assumed that, for large data sequences transmitted block-by-block, decoding can be usefully implemented only with the aid of a decoding window.

For reasons of data throughput and hardware cost the window MaxLogMAP algorithm is generally used for decoding.

Decoding results (soft output values) which are, by way of comparison, obtained using a MaxLogMAP algorithm without a decoding window may be more precise, but expensive hardware and memories are required for this purpose.

An alternative to the MaxLogMAP algorithm, based on a block error probability with a given signal-to-noise ratio, is provided by the "soft output Viterbi algorithm" (SOVA) described in DE 39 10 739 C3 and DE 42 24 214 C2. However, the SOVA algorithm has a smaller correlation between decoding errors and soft output values formed than the MaxLogMAP algorithm.

The object of the present invention is to carry out a decoding of a data sequence encoded with the aid of a binary convolutional code, using a MaxLogMAP algorithm, in such a manner that precise soft output values are formed as decoding results with inexpensive hardware.

The object of the invention is achieved by the features of Claim 1. Advantageous developments are indicated in the subclaims.

The method according to the invention is a processor-oriented implementation of the MaxLogMAP algorithm.

As a result of the memory cascading according to the invention on the one hand and the use of interpolation nodes for metrics calculation on the other hand, the MaxLogMAP algorithm can be efficiently integrated on exactly one application-specific module (ASIC).

The non-use of a decoding window must not entail a loss of accuracy of the decoding results.

According to the invention the metrics values are calculated productively in a first calculation operation and reproductively in further calculation operations based on the stored interpolation nodes.

The memory cascading according to the invention enables an optimal data throughput to be achieved.

The method according to the invention saves memory space and therefore area on an ASIC module. The area saved is therefore available

for further signal processing algorithms, which means that additional algorithms can be implemented in the ASIC module.

An exemplary embodiment of the invention is described below with reference to a drawing, in which:

FIG 1 is a diagrammatic representation of a MaxLogMAP algorithm for the precision calculation of soft output values according to the prior art,

FIG 2 is a diagrammatic representation of a window MaxLogMAP algorithm for the calculation of soft output values according to the prior art,

FIG 3 is a diagrammatic representation of a MaxLogMAP algorithm according to the invention for the precision calculation of soft output values, and

FIG 4 is an example of the metrics value calculation and storage according to the invention.

FIG 1 is a diagrammatic representation of a MaxLogMAP algorithm for the precision calculation of soft output values according to the prior art.

The MaxLogMAP algorithm is used to decode a data sequence which consists of K information bits and has been encoded with the aid of a binary convolutional code.

On a trellis diagram TREL, starting at a trellis segment T1, alpha metrics values Mα-calc-store are calculated and stored for every individual trellis segment TSN as logarithmic transition probabilities. At the same time, in parallel therewith, starting at a trellis segment T2, beta metrics values Mβ-calc-store are calculated and stored for every individual trellis segment TSN.

The two calculations pass each other at a trellis segment TSM, a decision process being carried out from this time onwards for the purpose of calculating a soft output value, i.e. an information bit of

the data sequence is decoded. When that happens, in the course of a "forward decision process" FDP, after the trellis segment TSM has been passed, currently calculated alpha metrics values Mα-calc are used with the previously calculated and stored beta metrics values Mβ-calc-store to calculate the soft output value.

This procedure takes place at the same time in a "backward decision process" BDP, wherein currently calculated beta metrics values Mβ-calc are used with the previously calculated and stored alpha metrics values Mα-calc-store to calculate the soft output value.

The following reference characters are used below:

K	number of information bits,
s	state of a convolutional code decoding,
m	code memory length,
T	required number of trellis segments, where T = K + m, and
π	length of a transient phase, where π > 5*m

With a word width w of a metrics memory and assuming that the respective metrics values are standardized, two metrics processors and a total of $2 \cdot K/2 \cdot w \cdot 2^m$ memory locations are required for this implementation of the MaxLogMAP algorithm. The data throughput achieved is as follows:

$$\frac{1}{t_{segment} \cdot 10^6} \text{ [Mbit/s]} ,$$

where the above parameter $t_{segment}$ is dependent on the module technology (ASIC) used to implement the algorithm, on the memory architecture and on the clock speed used with the ASIC module.

In the case of terminated codes, each calculation of the metrics values starts from the assumption of an uneven probability distribution in a trellis segment having an initial state with a probability of 100%, while all further states have a probability of 0%.

In the case of what are known as "tailbiting codes" no initial state is known at the outset. A transient phase of length π therefore has to be introduced for both directions. Associated metrics values for the transient phase are indicated as Mα-pre and Mβ-pre respectively.

FIG 2 is a diagrammatic representation of a window MaxLogMAP algorithm for the calculation of soft output values according to the prior art.

The window MaxLogMAP algorithm, which is implemented with the aid of a sliding decoding window, is used for long data sequences. The particular advantage of the window MaxLogMAP algorithm is its efficient implementation.

Similarly to FIG 1, alpha metrics values Mα-calc are calculated precisely in a forward direction starting at a trellis segment T1. By contrast, beta metrics values Mβ-calc-1 of a first decoding window DP1 and beta metrics values Mβ-calc-2 of a second decoding window DP2 are estimated.

If the two decoding windows DP1 and DP2 reach the right-hand edge of the trellis diagram TREL, all the termination information is available. The beta metrics values Mβ-calc-1 and Mβ-calc-2 are calculated precisely, and the corresponding soft output values are formed.

Metrics values Mα-pre, Mβ-pre-1 and Mβ-pre-2 are again assigned to a transient phase.

The window MaxLogMAP algorithm is described in detail in the above-mentioned publication "An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes".

For this implementation, with a data throughput similar to that of FIG 1, in total $\lceil w/\psi \rceil + 1$ metrics processors and $(1+\theta)\cdot\psi\cdot w\cdot 2^m$ storage locations are required,

where:

ψ is the number of trellis segments for which 2^m metrics are stored in each case,

w is the size of the decoding window in trellis segments, and

$\theta \in \{0,1\}$ is a parameter which is dependent on the module technology (ASIC) used to implement the algorithm, on the memory architecture and on the clock speed used with the ASIC module.

Especially where high speed channels have code rates close to one (generated with the aid of a points system), use of the decoding window leads to unacceptable deteriorations in performance, however.

FIG 3 is a diagrammatic representation of a MaxLogMAP algorithm according to the invention for the precision calculation of soft output values.

Similarly to FIG 1, alpha metrics values are again calculated in a forward direction and beta metrics values in a backward direction of each trellis segment TSN. However, now, in accordance with the invention, only metrics values of a selected number of trellis segments which act as interpolation nodes are selected and stored.

In a preferred embodiment they are stored in a memory divided into levels in a cascaded manner.

The following reference characters are used below:

m code memory length,

sm	memory length shift of a feed-forward terminated code (in the case of a recursively terminated code and in the case of a tailbiting code sm = 0),
K	number of information bits,
T	required number of trellis segments, where T = K + m;
π	length of the transient phase where $\pi > 5*m$,
$\delta(1)$	memory depth of a first memory level SP(1) for interpolation nodes of a first metrics value calculation,
$\delta(n-1)$	memory depth of an n-1-th memory level SP(n-1) for interpolation nodes of an n-1-th metrics value calculation, and
$\delta(n)$	memory depth of an n-th memory level SP(n) for interpolation nodes of an n-1-th metrics value calculation.

On the trellis diagram TREL, in a first operation, starting at a trellis segment T1, alpha metrics values $M\alpha$-calc(1) are calculated in a forward direction FDP and, starting at a trellis segment T2, beta metrics values $M\beta$-calc(1) are calculated in a backward direction BDP for each one of the trellis segments TSN as logarithmic transition probabilities.

According to the invention, however, from the calculated metrics values $M\alpha$-calc(1) and $M\beta$-calc(1) of the first operation, metrics values $M\alpha$-calc-sel(1) and $M\beta$-calc-sel(1) of the first operation are each now filed in a first memory level SP(1) with a memory depth of $\delta(1)$ for a selection of $K/\delta(1)$ trellis segments acting as interpolation nodes.

In a second operation, on the basis of each pair of adjacent interpolation nodes of the first operation, metrics values $M\alpha$-calc(2) and $M\beta$-calc(2) are calculated for the trellis segments TSN positioned between the respective interpolation nodes of the first operation.

As in the first operation, from the calculated metrics values Mα-calc(2) and Mβ-calc(2) of the second operation, for a selection of K/δ(1)/δ(2) trellis segments again acting as interpolation nodes the corresponding metrics values Mα-calc-sel(2) and Mβ-calc-sel(2) of the second operation are filed in a second memory level SP(2) with a memory depth of δ(2).

This metrics value calculation based on the interpolation nodes of a previous operation is accordingly continued both in a forward direction and in a backward direction. During this process, after the trellis segment TSM is passed, corresponding soft output values are formed, memory levels that are released being accordingly reused.

Here the decision process for the determination of the soft output values is as described in FIG 1. The individual memory levels are arranged in a mutually cascaded structure.

After n operations all soft output values are determined, the n-th memory level having a memory depth of δ(n) with stored metrics values of K/δ(1)/δ(2)/..../ δ(n) trellis segments or interpolation nodes.

Comparing this with the data throughput stated in FIG 1 and FIG 2, the method according to the invention requires a total of $2 \cdot n$ metrics processors and $\left(\sum_{i=1}^{n}(1+\theta_i)\cdot\delta_i\right)\cdot w\cdot 2^m$ storage locations.

The following applies: $\prod_{i=1}^{n}\delta_i = K$, $\theta_i \in \{0,1\}$, according to the number of ASIC clock cycles required for one trellis segment and the number of ports available on the memories.

On the basis of existing implementations of the MaxLogMAP window algorithm, parameters can be deduced and can be used for a comparison

between the conventional implementations known from FIG 1 and FIG 2 and the implementation according to the invention.

The results can be found in the following Table:

	Logic [Kgates]	Memory [Kbit]	"Area" [Kgates]
MaxLogMAP	101	1080	2300
Window MaxLogMAP	233	138	500
Memory-cascaded implementation	233	106	450

With the same data throughput and with no loss of accuracy in the MaxLogMAP algorithm as a result of the non-use of a decoding window, hardware costs for the memory-cascaded implementation according to the invention are reduced by over 80% compared with the conventional implementation described in FIG 1.

The following assumptions were made for this comparison:

- an overhead resulting from soft input and soft output buffer at a systems interface with $\left(R^{-1}+1\right)\cdot w_{soft}\cdot k\cdot a_{memory}$
  where k=K generally applies but $k=W+((n_{\xi}-1)*\delta)$ may apply to implementation with decoding window; where R is code rate, w_{soft} is word width of the soft values and a_{memory} is unit area per memory bit,
- an overhead resulting from control module and interface: $A_{overhead}$.
- a metrics processor including scaling of the register bits: $A_{viterbi}(w)\cdot f_{parallel}+2^{(m+1)}\cdot w\cdot a_{flipflop}$
  where $A_{viterbi}$ represents the Viterbi arithmetic for a given word width w and $a_{flipflop}$ represents the register unit area per bit, $f_{parallel}$ is the number of butterfly calculations made in an ASIC clock cycle and can assume the following values: $\{1,2,\ldots,2^{(m-1)}\}$,
- a metrics word width $w = 16$,
- a soft value word width $w_{soft} = 8$,
- a memory bit area $a_{memory} = 2$,

- a register bit area $a_{flipflop}$ = *10* units,
- a Viterbi arithmetic/butterfly $A_{viterbi}$ = *12500* units,
- an overhead $A_{overhead}$ = *50000* units.

The units correspond to a gate equivalent of a 0.18 µm ASIC module at a clock frequency of approx. 150 MHz.

Data from a parameter set relevant to the GSM/EDGE standard are compared below:

- code rate R = 1/6
- memory length m = 6
- block size K = 1000
- decoder throughput greater than 2 Mbit/s
- parallelism $f_{parallel}$ = 1

	Configuration (@ 2.6 Mbit/s)	"Area" [Kgates]
Exact implementation	2 metrics processors	2250
Window implementation (path fusion limit = 160) window also with soft input and soft output memory	7 metrics processors (1+θ) metrics memory with δ=32, θ=1 1 soft input and soft output memory with window	480
Window implementation (path fusion limit = 160)	6 metrics processors (1 + θ) metrics memory with δ=40, θ=1 1 soft input and soft output memory	500
Window implementation (path fusion limit = 160)	5 metrics processors (1+θ) metrics memory with δ=54, θ=0 1 soft input and soft output memory	425
Memory-cascaded imple-mentation	6 metrics processors 2 interpolation nodes memory at the 1st level with δ_1=18; (1+θ_2)*2 interpolation nodes memory at the 2nd level with δ_2=8, θ_2=1; (1+θ_3)*2 metrics memories with δ_3=7, θ_3=1; 1 soft input and soft output memory	450

	Configuration (@ 2.6 Mbit/s)	"Area" [Kgates]
Memory-cascaded implementation	6 metrics processors 2 interpolation nodes memories at the 1st level with δ_1=10; $(1+\theta_2)$*2 interpolation nodes memory at the 2nd level with δ_2=10, θ_2=0; $(1 + \theta_3)$ * 2 metrics processors with δ_3=10, θ_3=0; 1 soft input and soft output memory	400

In the case of UMTS (W-CDMA) and with UTRAN TDD convolutional codes, the following parameters are to be used for the convolutional decoding:

- code rate R = 1/3
- memory length m = 8
- maximum block size K = 300
- decoder throughput greater than 2 Mbit/s
- parallelism $f_{parallel}$ = 8

	Configuration (@ 3.125 Mbit/s)	"Area" [Kgates]
Exact implementation	2 metrics processors	2891
Window implementation with sliding window on soft input / soft output memory	6 metrics processors δ=40, θ=1	1848
Memory-cascaded implementation	4 metrics processors δ_1= 20, δ_2 = 15 θ_1 = 0, θ_2 = 1	1206

In the case of UMTS (W-CDMA) and with UTRAN TDD turbo codes a slightly expanded MaxLogMAP decoder can be used as part of the

turbo-decoding. This means that, here too, memory-cascaded implementation can be compared with direct and window implementation:

- code rate $R = 1/3$
- memory length $m = 3$
- maximum block size $K = 5200$
- decoder throughput greater than *2 Mbit/s*
- parallelism $f_{parallel} = 1/4$

	Configuration (@3.125 Mbit/s)	"Area" [Kgates]
Exact implementation	2 metrics processors	1727
Window implementation with sliding window also for soft input/soft output memory	6 metrics processors δ=40, θ=1	159
Memory-cascaded implementation	8 metrics processors δ_1=13; δ_2=10, δ_3=8, δ_4=5 $\theta_1 = 0$, $\theta_i = 1$ for $i = \{2,3,4\}$	448

FIG 4 shows an example of metrics value calculation and storage according to the invention.

In a first operation D1, alpha metrics values and beta metrics values are calculated. For each sixth trellis segment TSN = 1, 7, 13, ..., 73 acting as interpolation point, 2^m calculated alpha metrics values are stored in a memory level SP(α1) and 2^m calculated beta metrics values are stored in a memory level SP(β1).

In a second operation D2, these metrics values are read out of the memory levels SP(α1) and SP(β1) and, for the trellis segments positioned between the interpolation nodes, the associated alpha metrics values and beta metrics values are calculated precisely. Relevant

trellis segments are again selected as interpolation nodes, and the associated metrics values are stored. Two memory levels SP(α2) and SP(α2'), and SP(β2)and SP(β2'), respectively, are indicated here by way of example.

In a third operation D3, the trellis segment TSM is reached from both sides, and currently calculated 2^m beta metrics values with stored 2^m alpha metrics values filed in the memory SP(α2) are used for the backward decision process to determine soft output values. In exactly the same way, currently calculated 2^m alpha metrics values with stored 2^m beta metrics values filed in the memory SP(β2) are used for the forward decision process to determine soft output values.

After a total of n=3 operations Dn all the decisions values $M_{decisions}$ were formed.